What is claimed is:

1. A method for operating a processor comprising:
in response to executing a single arithmetic instruction,
multiplying a first number by a second number; and
adding implicitly a partial result from a previously executed single arithmetic instruction to generate a result that represents the first number multiplied by the second number summed with the partial result.

2. The method as recited in claim 1 further comprising performing the adding of the partial result as part of addition operations performed for the multiplying of the first and second number.

3. The method as recited in claim 1 wherein the partial result is in redundant number representation.

4. The method as recited in claim 1 further comprising performing the adding of the partial result by adding the partial result to a multiplication result of the first and second numbers.

5. The method as recited in claim 1 further comprising storing a high order portion of the result as a next partial result for use with execution of a subsequent single arithmetic instruction.

6. The method as recited in claim 5 further comprising
storing the high order portion of the result into an extended carry register for use with execution of the subsequent single arithmetic instruction.

7. The method as recited in claim 6, further comprising retrieving an indication of a current value of the extended carry register by executing another single arithmetic instruction that multiplies a third number by a fourth number; and implicitly adds current contents of the extended carry register to generate a second result that represents the third number multiplied by the fourth number summed with the current contents of the extended carry register.

8. The method as recited in claim 7, wherein a low order portion of the second result contains the indication of the current value of the extended carry register.

9. The method as recited in claim 7, wherein the third and fourth numbers are zero.

10. The method as recited in claim 6, further comprising loading the extended carry register with a predetermined value by executing another single arithmetic instruction that multiplies a third number by a fourth number; and implicitly adds a current value of the extended carry register, to generate a result that represents the third number multiplied by the fourth number summed with the current value of the extended carry register, thereby loading the extended carry register with the predetermined value.

11. The method as recited in claim 6, further comprising selecting one of a plurality of extended carry registers as the extended carry register.

12. The method as recited in claim 6, further comprising accessing the extended carry register by at least one of a load and store instruction executed by the processor.

13. The method as recited in claim 1 wherein the first and second numbers are specified in the single multiply-accumulate instruction as first and second source registers and a low order portion of the result is stored in a destination location specified in the single multiply-accumulate instruction.

14. The method as recited in claim 5 wherein the first and second numbers are n-bit numbers, n being a positive integer and wherein the high order portion of the addition result is n bits.

15. The method as recited in claim 5 further comprising:
in response to executing the subsequent single arithmetic instruction, multiplying third and fourth numbers specified by the subsequent single arithmetic instruction and adding implicitly the next partial result to

generate a second result that represents the third number multiplied by the fourth number summed with the next partial result.

16. The method as recited in claim 15 as recited further comprising storing the high order portion of the second result to be implicitly added with another subsequent single multiply-accumulate instruction.

17. The method as recited in claim 1 wherein the multiplying and adding are implemented to support XOR operations for binary polynomial fields.

18. A method for operating a processor comprising:
in response to executing a single arithmetic instruction,
multiplying a first number by a second number;
adding implicitly a partial result from a previously executed single arithmetic instruction; and
adding a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number.

19. The method as recited in claim 18 further comprising performing the adding of the partial result as part of addition performed for the multiplying of the first and second number.

20. The method as recited in claim 18 wherein the partial result is stored in a redundant number representation.

21. The method as recited in claim 18 further comprising performing the adding of the third number as part of the addition performed for the multiplying of the first and second number.

22. The method as recited in claim 18 further comprising performing the adding of the partial result by adding the partial result after generation of a multiplication result of multiplying the first and second numbers.

23. The method as recited in claim 18 further comprising storing a high order portion of the result as a next partial multiplication result for use with execution of a subsequent single arithmetic instruction.

24. The method as recited in claim 23 further comprising
storing the high order portion of the result into an extended carry register for use with execution of the subsequent arithmetic instruction.

25. The method as recited in claim 24, further comprising retrieving an indication of a current value of the extended carry register by executing another single arithmetic instruction that multiplies a fourth number by a fifth number; and implicitly adds current contents of the extended carry register and adds a sixth number to generate a second result that represents the fourth number multiplied by the fifth number summed with the current contents of the extended carry register and the sixth number.

26. The method as recited in claim 25, wherein a low order portion of the second result contains the indication of the current value of the extended carry register.

27. The method as recited in claim 24, further comprising loading the extended carry register with a predetermined value by executing another single arithmetic instruction that multiplies a fourth number by a fifth number; and implicitly adds a current value of the extended carry register and adds a sixth number, to generate a result that represents the third number multiplied by the fourth number summed with the current value of the extended carry register and summed with the sixth number, thereby loading the extended carry register with the predetermined value.

28. The method as recited in claim 24, further comprising selecting one of a plurality of extended carry registers as the extended carry register.

29. The method as recited in claim 24, further comprising accessing the extended carry register via at least one of a load and store instruction executed by the processor.

30. The method as recited in claim 24 wherein the second number is implicitly identified in the single arithmetic instruction.

31. The method as recited in claim 30 further comprising accessing a special register storing the second number via at least one of a load and store instruction executed by the processor.

32. The method as recited in claim 18 further comprising the processor accessing a special register storing the second number via at least one of a load and store instruction.

33. The method as recited in claim 18 wherein the first and third numbers are specified in the single arithmetic instruction as first and second source registers and a low order portion of the result is stored in a destination location specified in the single arithmetic instruction.

34. The method as recited in claim 33 wherein the partial result from a previously executed single arithmetic instruction is implicitly specified by the single arithmetic instruction and wherein the second number is explicitly specified by a third source register in the single arithmetic instruction.

35. The method as recited in claim 33 wherein the partial result from a previously executed single arithmetic instruction is implicitly specified by the single arithmetic instruction and the second number is implicitly specified by the single arithmetic instruction.

36. The method as recited in claim 23 further comprising:
in response to executing the subsequent single arithmetic instruction, multiplying a fourth number and a fifth number, the fourth number being specified by the subsequent single arithmetic instruction and adding

implicitly the next partial multiplication result and adding a sixth number to generate a second result, the second result representing the fourth number multiplied by the fifth number summed with the next partial result and the sixth number.

37. The method as recited in claim 36 wherein the fifth number and the second number are equal.

38. The method as recited in claim 36 further comprising storing the high order portion of the second result to be implicitly added with another subsequent single arithmetic instruction.

39. The method as recited in claim 18 wherein the first number is specified in the single arithmetic instruction in a first source register and the second number is contained in a special register and is not specified in the single arithmetic instruction, and the third number is specified as a second source register in the single arithmetic instruction and a low order portion of the result is stored in a destination location specified in the single arithmetic instruction.

40. The method as recited in claim 18 wherein the first, second, and third numbers are specified by source operands in the single arithmetic instruction.

41. The method as recited in claim 18 wherein one of the first, second, and third numbers and a destination location are specified by one operand in the single arithmetic instruction.

42. The method as recited in claim 18 wherein the multiplying and adding operations are implemented for binary polynomial fields.

43. A processor comprising an arithmetic circuit, the processor responsive to execution of a single arithmetic instruction to cause the arithmetic circuit to multiply a first and second number and add implicitly a high order portion of a partial result from a previously executed single arithmetic instruction, thereby generating a result

that represents the first number multiplied by the second number summed with the high order portion of the partial result.

44. The processor as recited in claim 43 further responsive to the single arithmetic instruction to store a high order portion of the result into an extended carry register for use with execution of a subsequent single arithmetic instruction.

45. The processor as recited in claim 43 wherein the high order portion of the previously executed single arithmetic instruction is stored in a redundant number representation.

46. The processor as recited in claim 45, wherein the extended carry register is a register accessible via a processor instruction.

47. The processor as recited in claim 45, wherein the extended carry register has an associated dirty bit indicating whether contents of the extended carry register need to be saved on the context switch.

48. The processor as recited in claim 43, wherein the extended carry register is a special register.

49. The processor as recited in claim 43 wherein the first and second numbers are specified in the single arithmetic instruction as first and second source registers.

50. A processor comprising an arithmetic circuit the processor responsive to a single arithmetic instruction that upon execution thereof causes the arithmetic circuit to multiply a first number and a second number and add a third number and implicitly add a high order portion of a previous result from a previously executed single arithmetic instruction thereby generating a result that represents the first number multiplied with the second number, summed with the high order portion of the previous result and with the third number.

51. The processor as recited in claim 50 wherein the processor is coupled to store a high order portion of the result for use with execution of a subsequent single arithmetic instruction.

52. The processor as recited in claim 50 wherein the high order portion of the previous result is stored in a redundant number representation.

53. The processor as recited in claim as recited in claim 50 wherein the processor is coupled to store the high order portion of the result into an extended carry register.

54. The processor as recited in claim 50, wherein the extended carry register is a special register accessible by the processor via at least one of load and store instructions.

55. The processor as recited in claim 50, wherein the extended carry register has an associated dirty bit indicating whether contents of the extended carry register need to be saved on a context switch.

56. The processor as recited in claim 50, wherein the first number is specified in the single arithmetic instruction as a first source register and the second number is contained in a logically local register and is not specified in the single arithmetic instruction, and the third number is specified as a second source register in the single arithmetic instruction and a low order portion of the result is stored in a destination location specified in the single arithmetic instruction.

57. A computer program product encoded on computer readable media, the computer program product comprising:

a single arithmetic instruction causing a processor executing the single arithmetic instruction to multiply a first number by a second number and implicitly add a high order portion of a previously executed single arithmetic instruction to generate a result that represents the first number multiplied with the second number and summed with a high order portion of a previously executed single arithmetic instruction, the

single arithmetic instruction further causing the processor executing the single arithmetic instruction to keep a high order portion of the result for use with execution of a subsequent single arithmetic instruction.

58. The computer program product as recited in claim 57 wherein the single arithmetic instruction includes a first and second source operand specifying the first and second number and a destination operand, the single arithmetic instruction causing the processor to store a low order portion of the result in a location specified by the destination operand.

59. The computer program product as recited in claim 57 further comprising the subsequent single arithmetic instruction causing the processor executing the subsequent single arithmetic instruction to multiply a third number by a fourth number and implicitly add the high order portion of the result.

60. The computer program product as recited in claim 59 further comprising another single arithmetic instruction causing the processor executing the other single arithmetic instruction to multiply a fifth number by a sixth number and to generate another result without implicitly adding another high order portion of another previously executed result and to store a high order portion of the other result for use with another subsequent single arithmetic instruction.

61. A computer program product encoded on computer readable media, the computer program product comprising a single arithmetic instruction causing a processor executing the single arithmetic instruction to:

multiply a first number by a second number;

add implicitly a partial multiplication result from a previously executed single arithmetic instruction and a third number to generate a result that represents the first number multiplied by the second number summed with the partial multiplication result and summed with the third number; and

store a high order portion of the result for use with execution of a subsequent single arithmetic instruction.

62. The computer program product as recited in claim 61 further comprising the subsequent second single arithmetic instruction causing the processor executing the subsequent single arithmetic instruction to multiply a fourth number by the second number and add a fifth number and implicitly add the high order portion of the result.

63. The computer program product as recited in claim 61 further comprising the subsequent single arithmetic instruction causing the processor executing the subsequent single arithmetic instruction to multiply a fourth number by a fifth number and add a sixth number and implicitly add the high order portion of the result.

64. A processor comprising:

means, responsive to a single multiply-accumulate instruction, for multiplying a first number with a second number and implicitly adding a partial result of a previously executed single multiply-accumulate instruction to generate a result that represents the first number multiplied by the second number summed with the partial result; and

means for storing a high order portion of the result for use with execution of a subsequent single multiply-accumulate instruction.

65. A processor comprising:

means, responsive to a single multiply-accumulate instruction, for multiplying a first number with a second number and implicitly adding a partial result of a previously executed single multiply-accumulate instruction, and for adding a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number; and

means for storing a high order portion of the result for use with execution of a subsequent multiply-accumulate instruction.